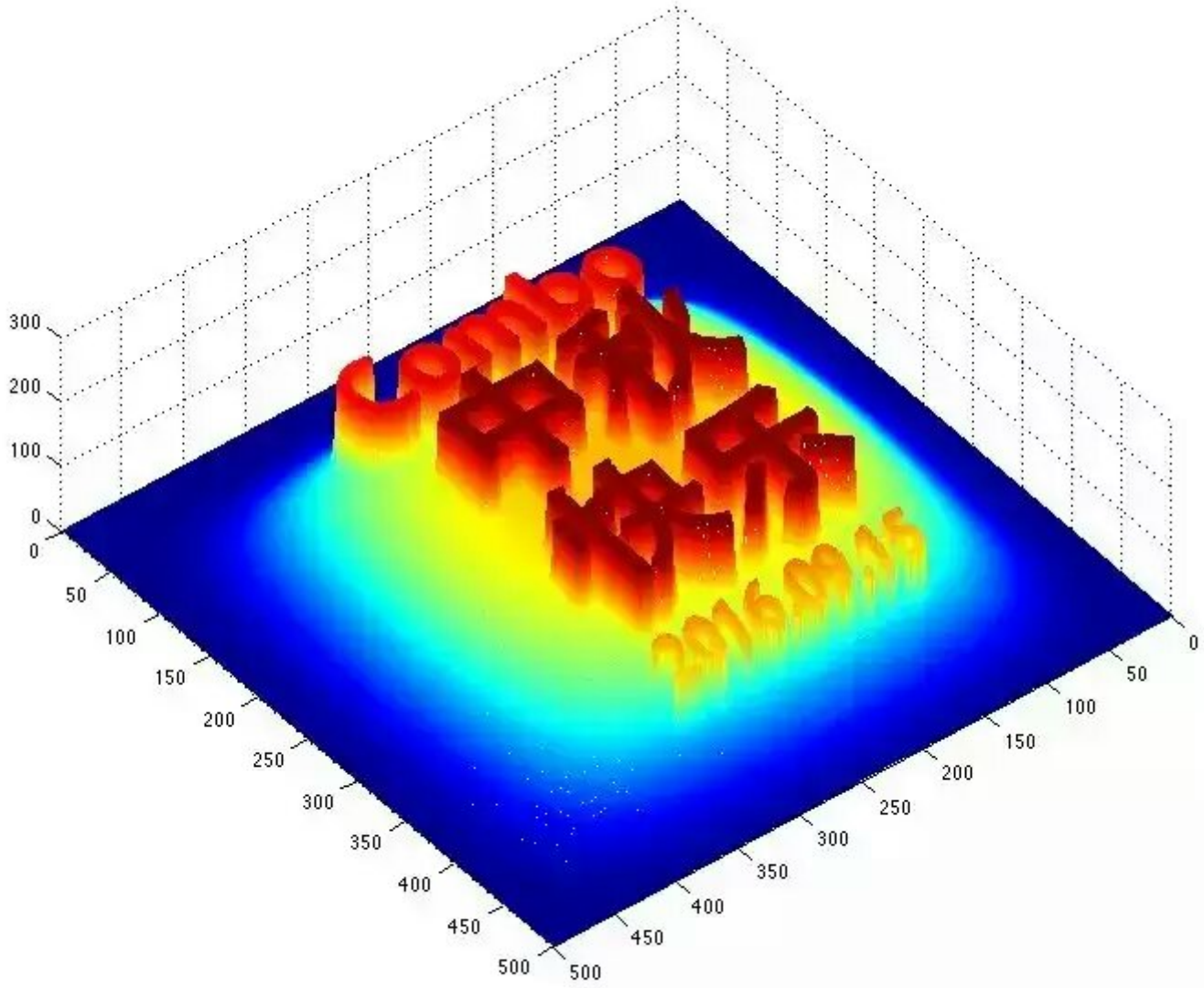
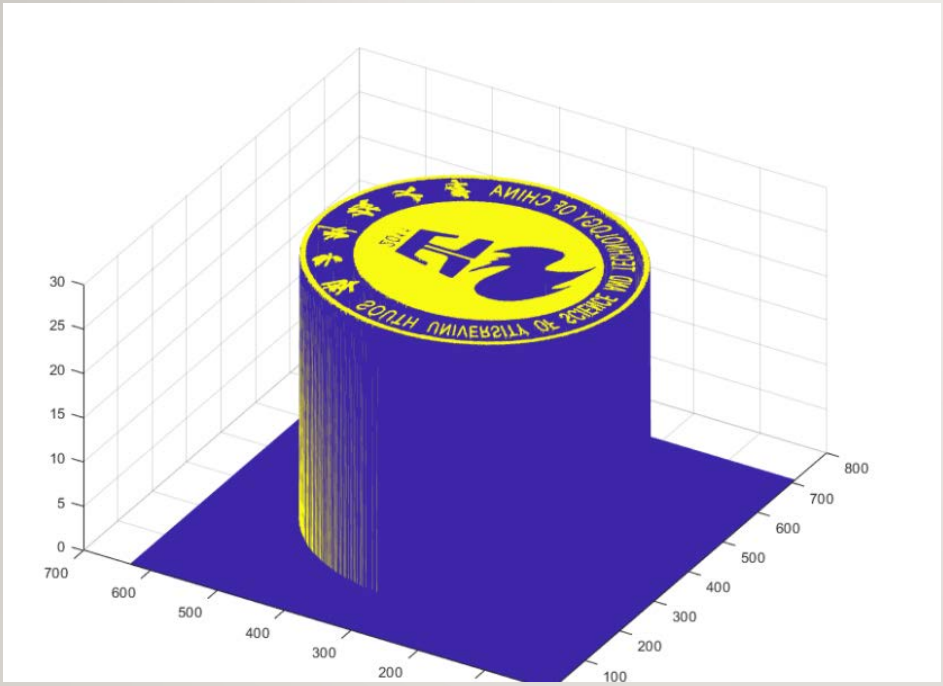
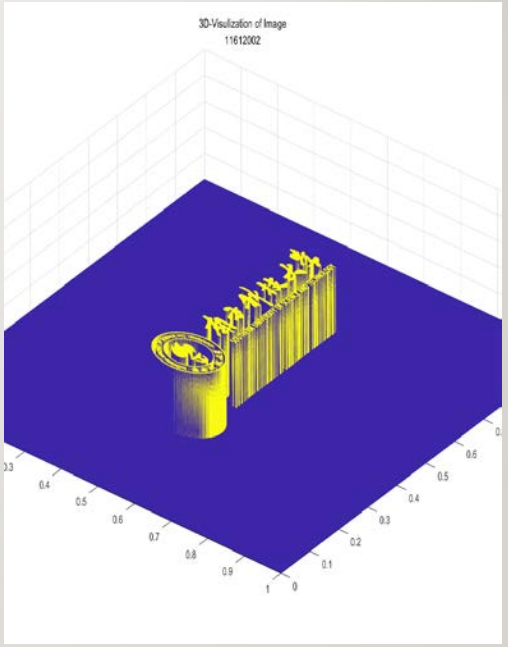
NONLINEAR OPTIMIZATION

Qingsha Cheng 程庆沙


300
200
100
0
0
50
100
150
200
250
300
350
400
450
500
500
450
400
350
300
250
200
150
100
50
0


30
25
20
15
10
5
0
700
600
500
400
300
200
800
700
600
500
400
300
200
100


3D-Visulization of Image
11612002

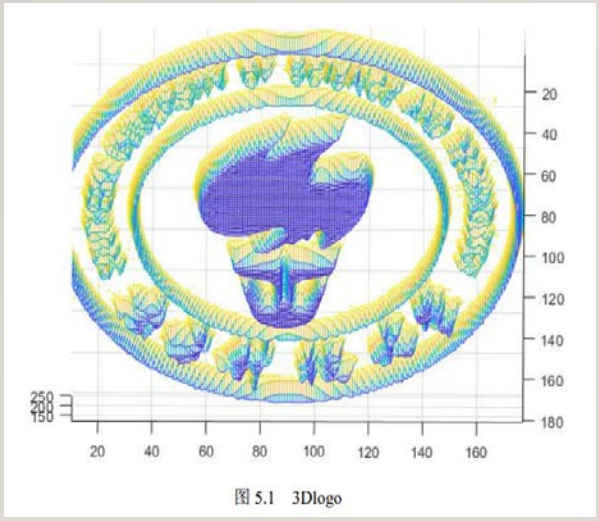
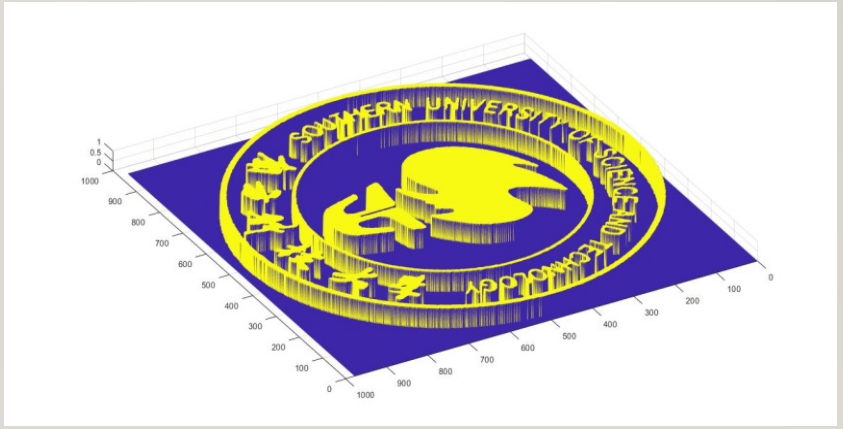
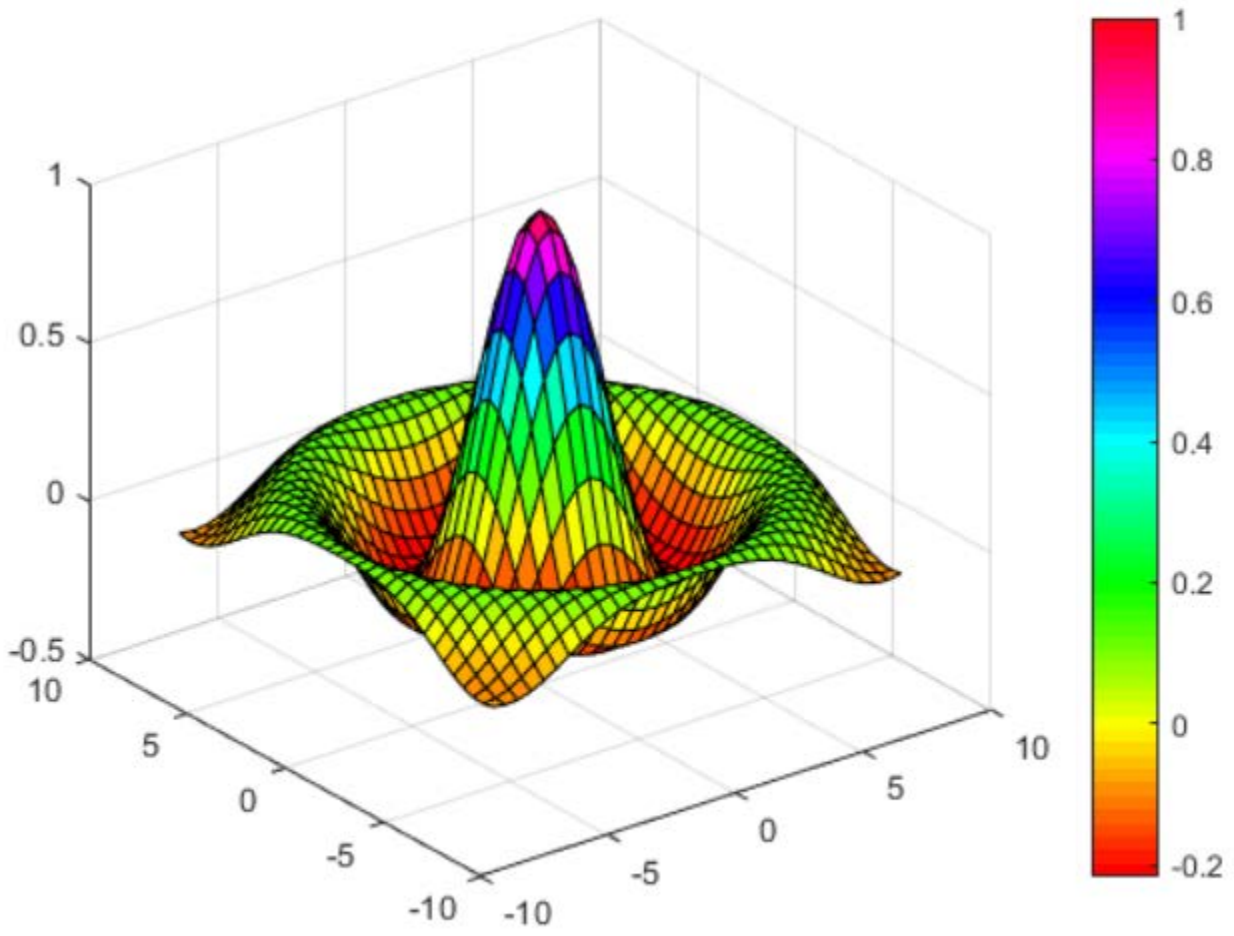

20
40
60
80
100
120
140
160
180
250
200
150
20
40
60
80
100
120
140
160

图 5.1 3Dlogo


1
0.5
0
1000
900
800
700
600
500
400
300
200
100
0
0
100
200
300
400
500
600
700
800
900
1000

```
[X,Y] = meshgrid(-8:.5:8);
R = sqrt(X.^2 + Y.^2) + eps;
Z = sin(R)./R;
surf(X,Y,Z)
colormap hsv
colorbar
```



MATLAB FUNCTION ARGUMENT

- For MATLAB built-in data types, MATLAB always passes arguments 'by value' in the sense that any changes made to input arguments within a function are not visible to the respective variables in the caller workspace.

```
clear all
a=1
dosomething(a);
a
```

```
function dosomething(a)
a=a+1;
end
```

MATLAB FUNCTION ARGUMENT

- We need to returned the modified argument as as an output argument of the function

```
clear all
a=1
a=dosomething(a);
a
```

```
function a=dosomething(a)
a=a+1;
end
```

MATLAB FUNCTION ARGUMENT

- However, since passing huge chunks of data 'by value' is inefficient, MATLAB internally optimizes for some cases and passes by reference and only creates local copies if modifications are being made to these arguments.

MATLAB FUNCTION ARGUMENT

- In the case of passing structures or cell arrays, **only the field or cell data being modified by the function will be passed "by value"**.
- When the function returns, (assuming that the modified structure or cell array is returned as an output argument of the function and are captured in the calling workspace's structure) the calling workspace's copy of the structure or cell array is replaced by the function's copy such that only the **changed** fields are altered.
- This is done to make copying more efficient. For example:

MATLAB FUNCTION ARGUMENT

```
% This will modify the structure in the calling
function's workspace.

largeStructure = myFunc(largeStructure);

% This will NOT modify the structure in the calling
function's workspace.

myFunc(largeStructure);
```

```
% Simple function
function [myStructure] = myFunc(myStructure)
    myStructure.x = myStructure.x^2;
end
```

Introduction to Engineering Optimization

Examples of engineering design optimization problems

Formulation of engineering optimization problem

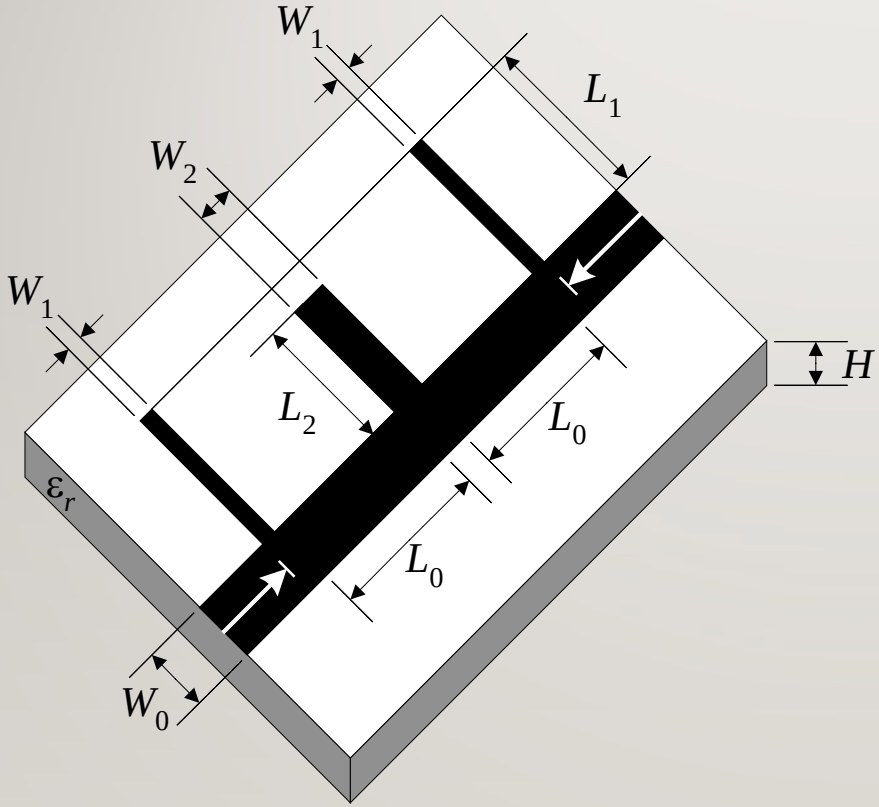
Merit functions and objective functions

Optimization process flow

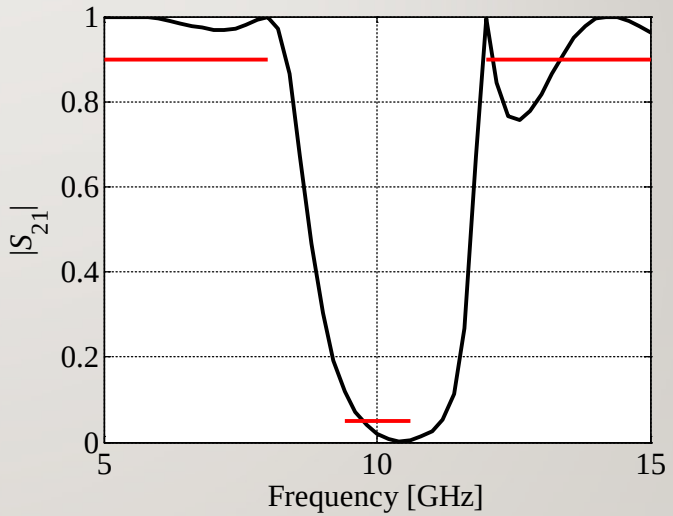
Classification of optimization methodologies

Choosing optimization method

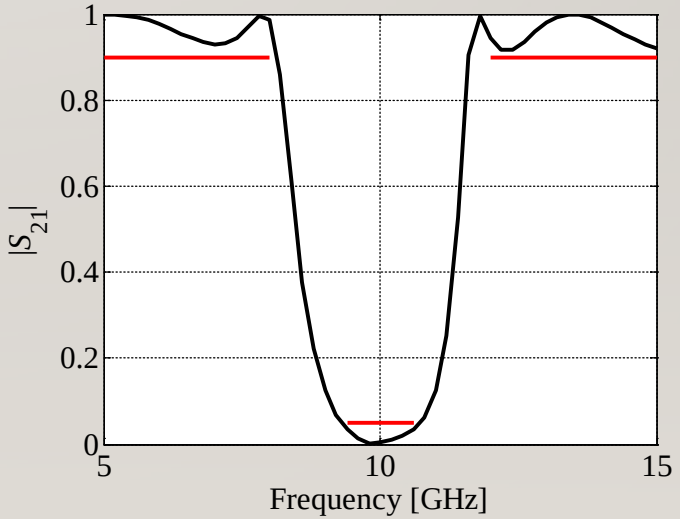
Example 1: Bandstop Microstrip Filter



Initial filter response



Optimized filter response



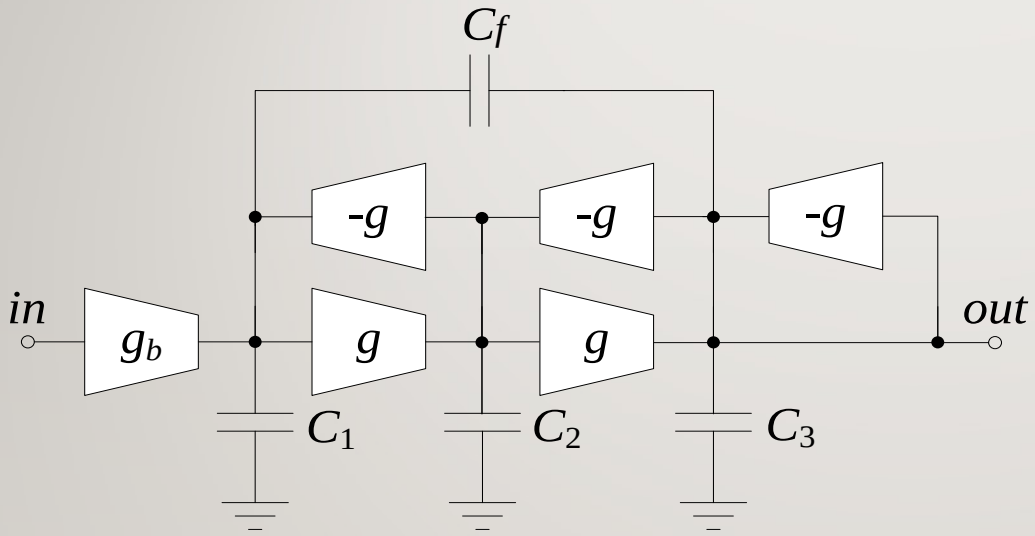
Design variables: $x = [W_1\ W_2\ L_0\ L_1\ L_2]^T$

Design specifications:

$|S_{21}| \leq 0.05$ for 9.4 GHz $\leq \omega \leq$ 10.6 GHz

$|S_{21}| \geq 0.9$ for $\omega \leq 8$ GHz and $\omega \geq 12$ GHz

Example 2: OTA-C* Elliptic Filter



Design variables:

$x = [g_b\ C_1\ C_2\ C_3\ C_f]^T$

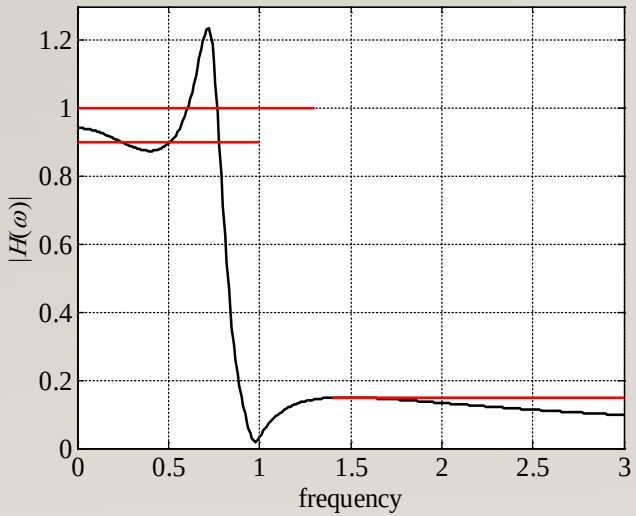
Design specifications:

$|H(\omega)| \geq 0.9$ for $\omega \leq 1.00$

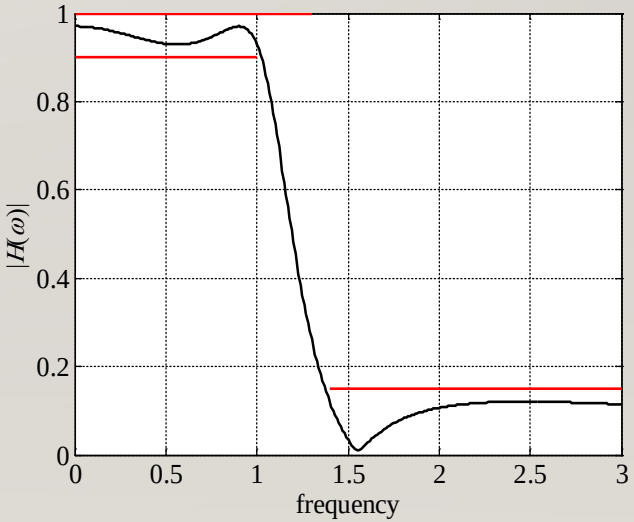
$|H(\omega)| \leq 1.0$ for $\omega \leq 1.30$

$|H(\omega)| \leq 0.15$ for $\omega \geq 1.40$

Initial filter response



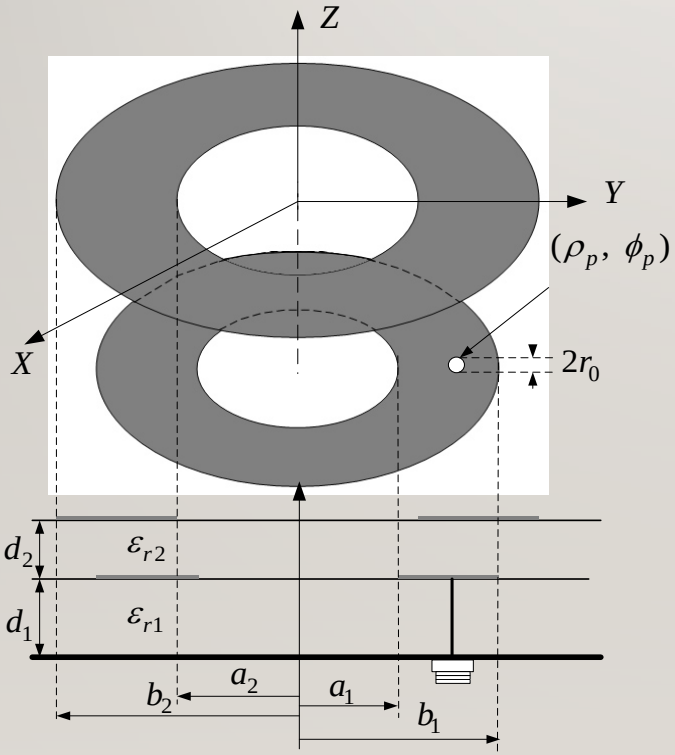
Optimized filter response



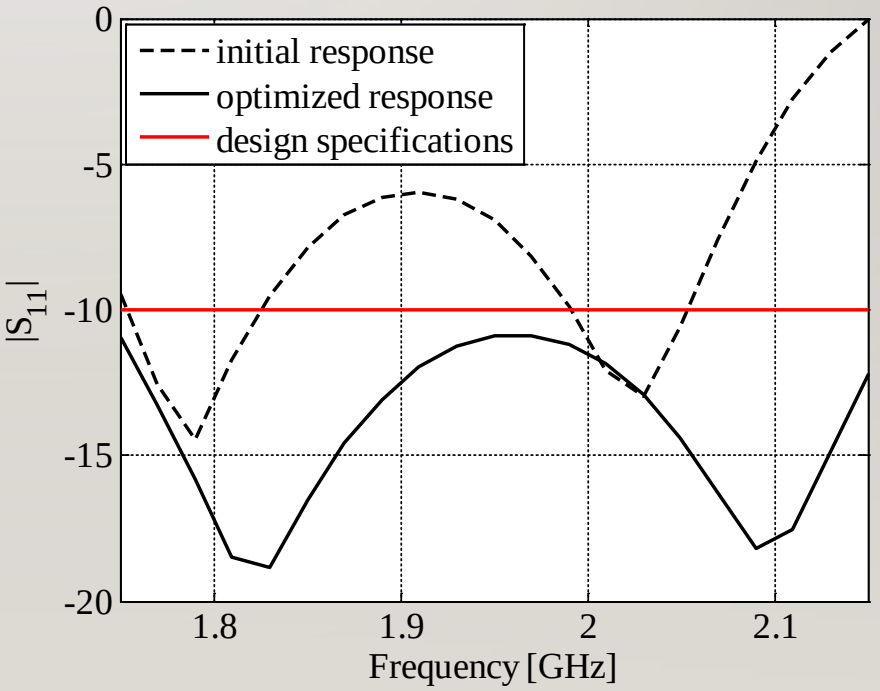
*Use Operational Transconductance Amplifiers (OTA) and Capacitors (C) only

Example 3: Antenna Design

Problem formulation: find geometrical dimensions a_1, a_2, b_1, b_2 and ρ of the double ring antenna so that the modulus of the reflection coefficient $|S_{11}|$ of the antenna is below -10dB within the frequency range 1.75 GHz to 2.15 GHz.



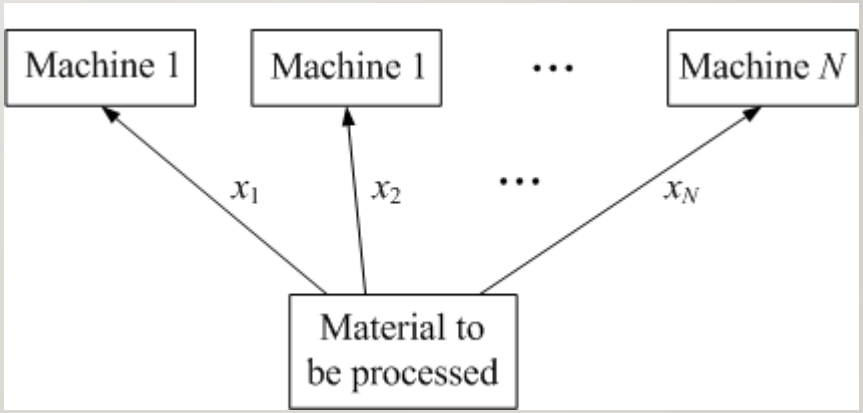
Antenna geometry



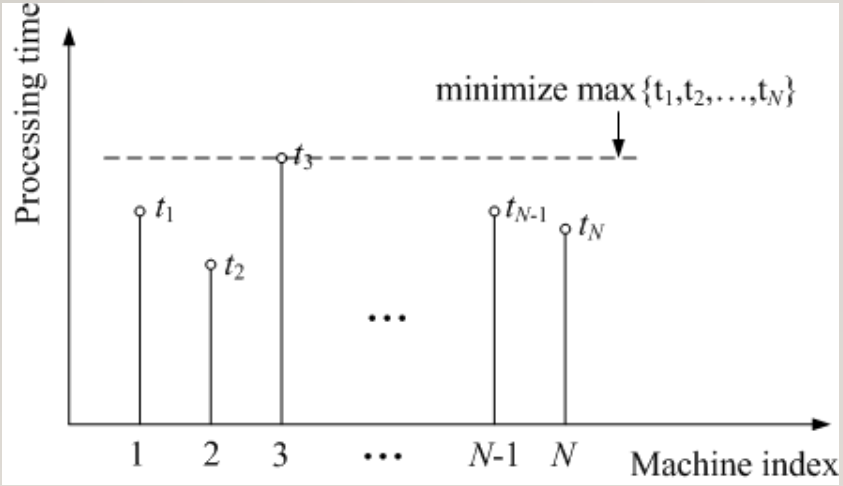
Initial and optimized antenna response

Example 4: Job Assignment Problem

Problem formulation: amount M of material is to be processed by N machines of efficiencies f_i, $i = 1, 2, \ldots, N$. Processing time t_i for each machine depends both on f_i and on x_i (the amount of material being processed on ith machine) so that $t_i = t_i(x_i, f_i)$.

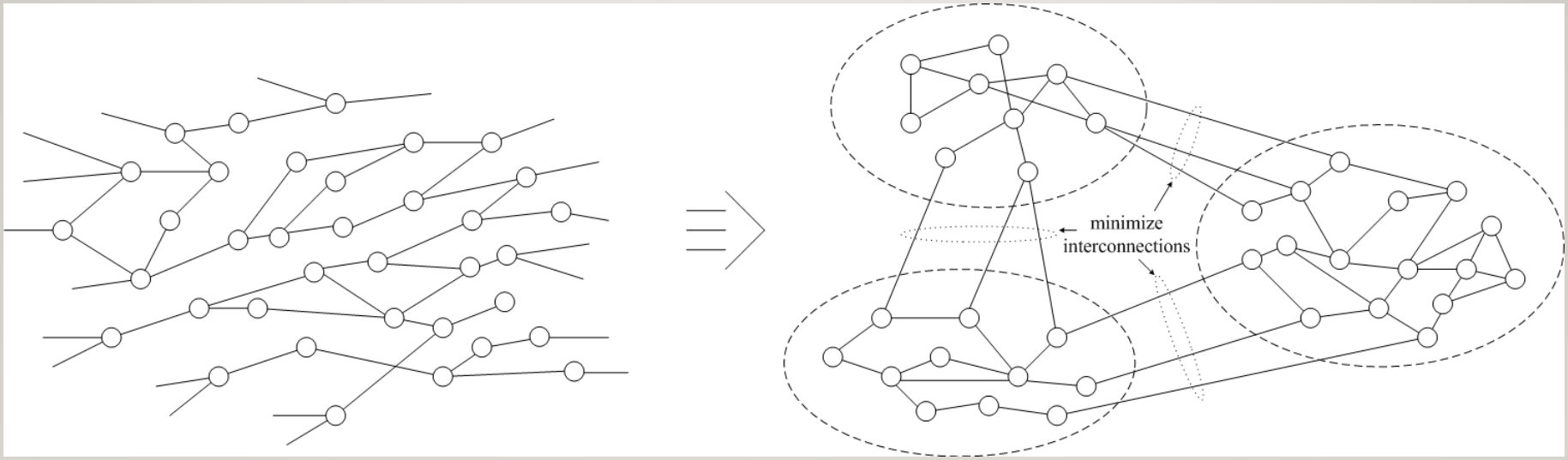


Goal: assign x_i so that the total processing time is minimized.



Example 5: VLSI Design – Circuit Partitioning

Problem formulation: divide the circuit (set of connected functional units into) *N* subsets so that the number of interconnections between the subsets is minimized.

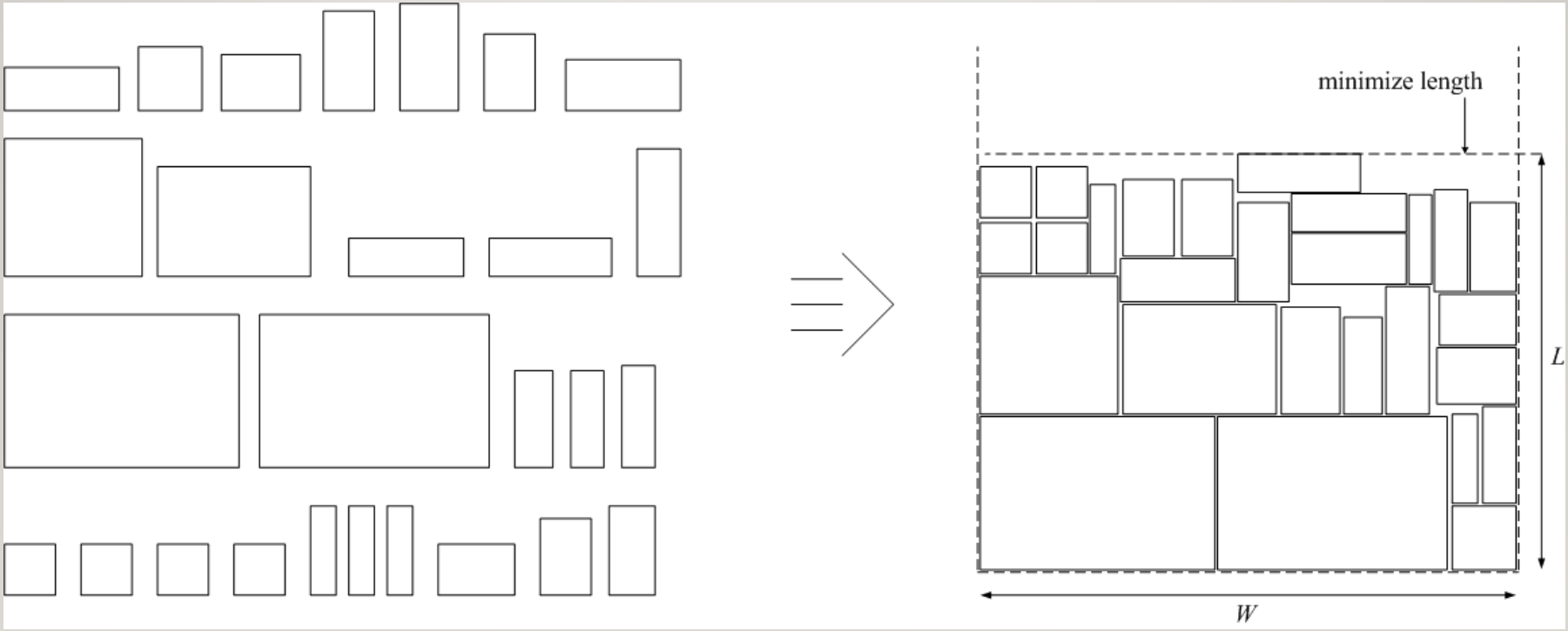


Optional constraints:

- maximum power consumption for each subset
- maximum area for each subset
- maximum number of subset inputs/outputs

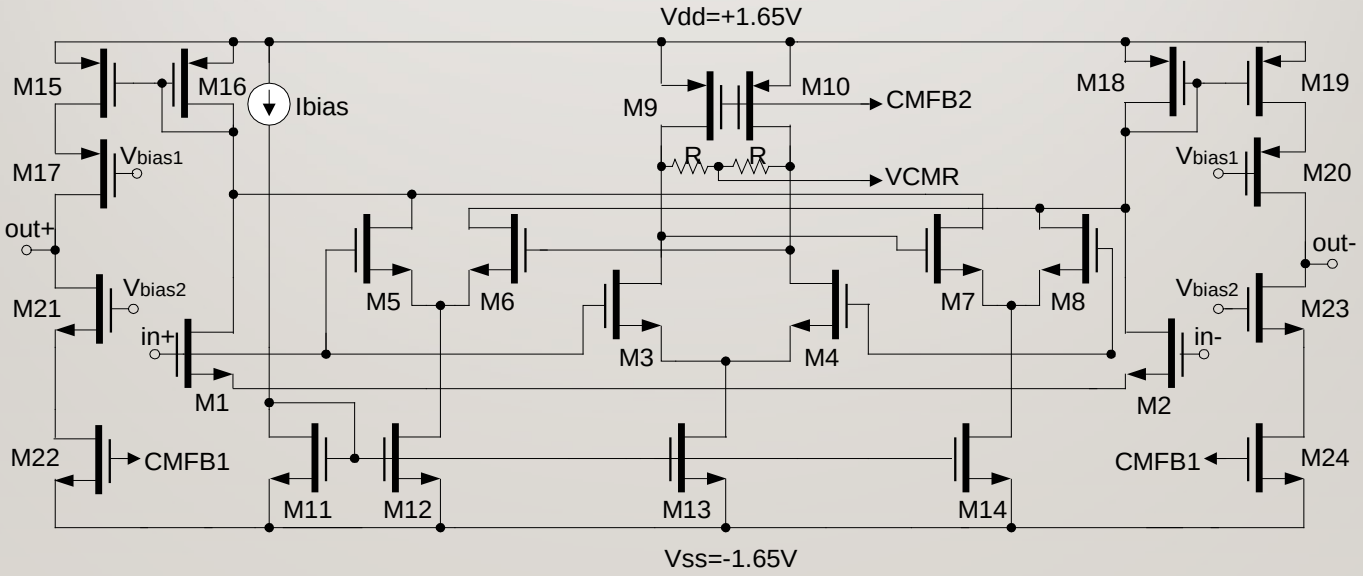
Example 6: VLSI Design – Placement

Problem formulation: place N rectangular elements of sizes $w_i \times l_i$, $i = 1,\ldots,N$, into rectangular bin of width W so that the length L of the space filled with the elements is minimized.



Example 7: Operational Transconductance Amplifier Design

Problem formulation: find dimensions (channel widths *W* and lengths *L*) of CMOS devices M1 to M24 so that amplifier gain and bandwidth are maximized, while its nonlinear distortion is minimized.



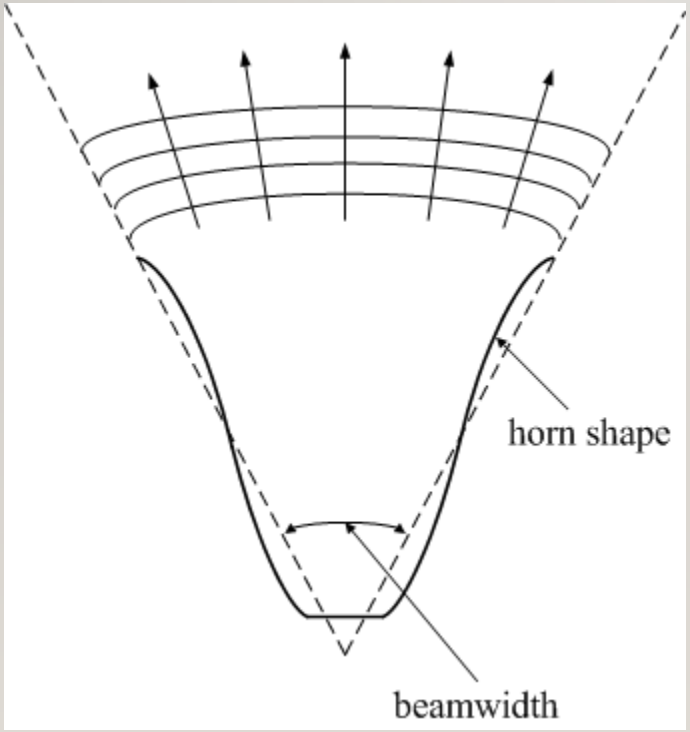
Optional constraints:

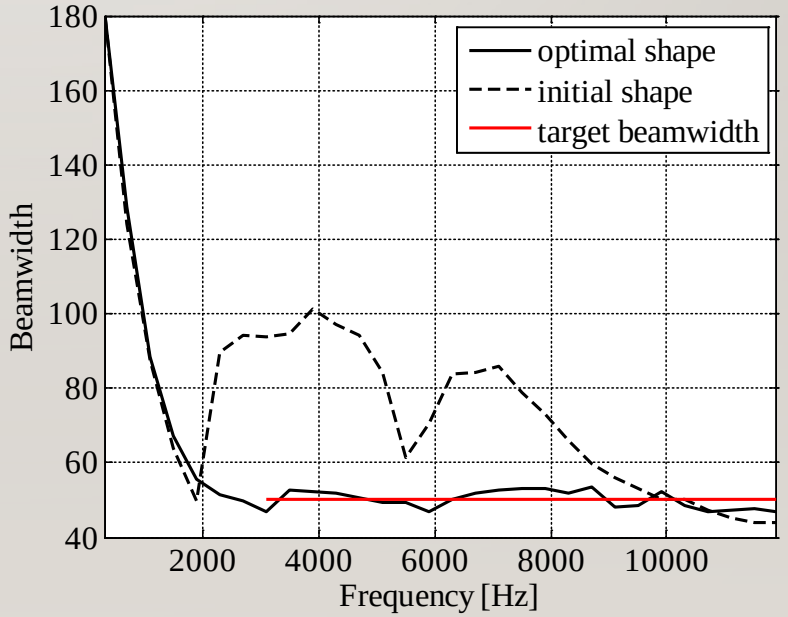
- maximum power consumption of the circuit
- maximum area of the circuit
- minimum/maximum *W*/*L* ratio

Example 8: Horn Shape Optimization

Problem formulation: find the optimal shape of the horn-loaded loudspeaker so that the sound is broadcast evenly onto the audience at all frequencies, i.e., obtain frequency independent beamwidth.

Experimental conical horn

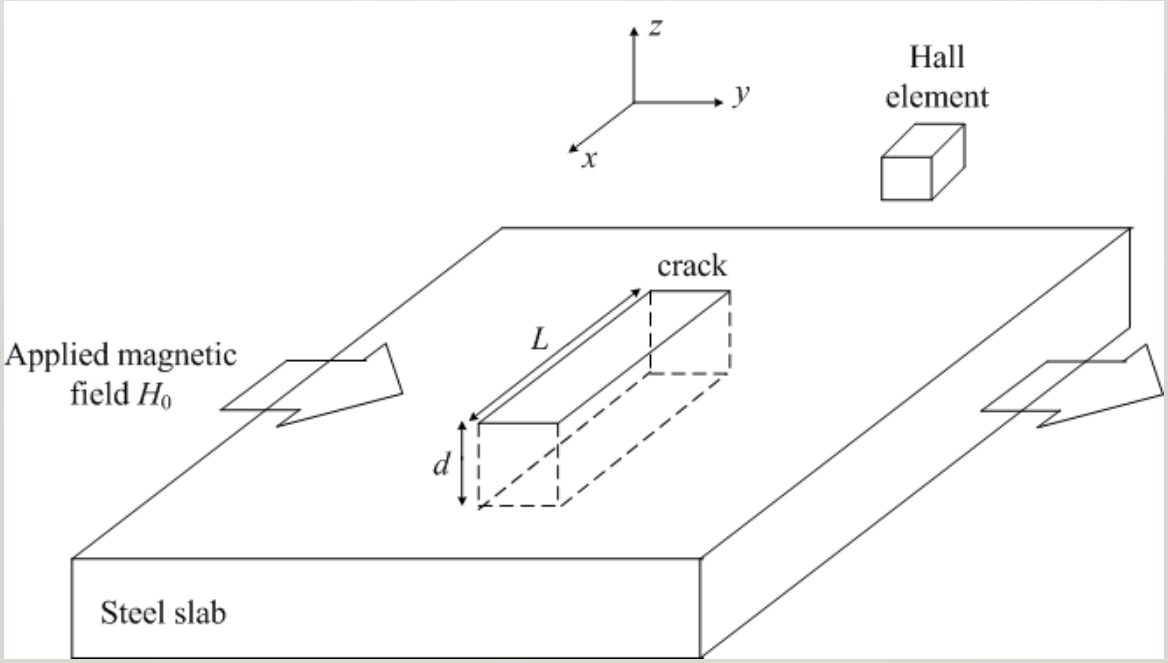




Initial and optimized beamwidth vs. frequency

Example 9: Crack Detection in Steel Pipes

Problem formulation: using measurements from a Hall element located over a magnetized steel slab estimate the depth *d* and length *L* of a crack shown in the picture below.



This is an example of a so-called inverse problem: given the system response for given set of parameters (here: crack dimensions) we need to retrieve the values of these parameters

Example 10: Traveling Salesman Problem (TSP)

Problem formulation: given a collection of cities and the cost of travel between each pair of them, find the cheapest way of visiting all the cities and returning to the starting point

Probably the most famous problem in computational mathematics

Visualize a TSP problem

A number of applications, e.g.:
- TSP naturally arises as sub-problem in many transportation and logistics tasks (e.g., arranging school bus routes, routing of trucks for parcel post pickup, scheduling collection of coins from payphones)
- scheduling of service calls at cable firms
- scheduling of a machine to drill holes in a circuit board
- solving sub-problems in genome sequencing

Example 10+: Tuning of Microwave Filter Using a Robot

Problem formulation: given a tunable microwave filter, use a robot to tune the screws so that filter responses are optimized.

Tuning Robot Video

This is an example of optimization algorithm used in fabrication process.

Solving Engineering Optimization Problem

1. Formulate the problem.
    A. Analytically
    B. Empirically
    C. Numerically

2. Identify design variables and design specifications.

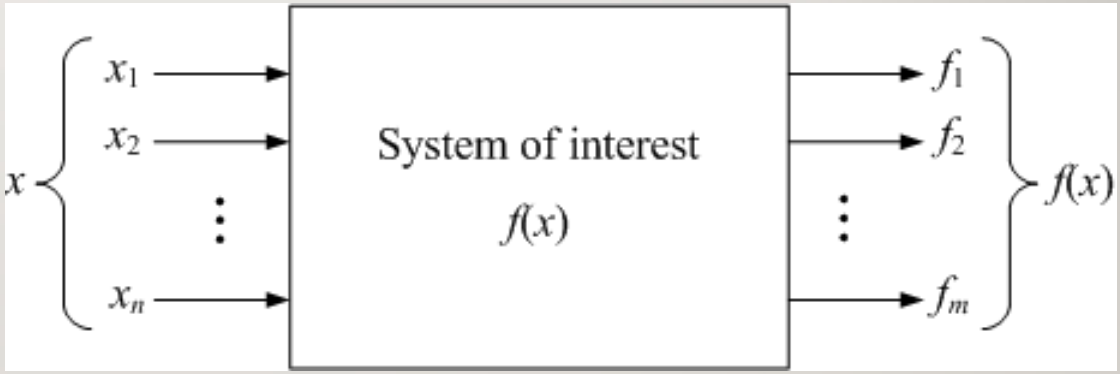
3. Define objective function and constraints.

4. Select optimization methodology.

5. Solve the problem.

Engineering Optimization Problem Formulation

Let $f: X \rightarrow R^m$, X Í R^n, denote a response of a system of interest, e.g., antenna or filter



Notation:

R – set of real numbers

$x = [x_1\ x_2\ \ldots\ x_n]^T$ Î X – design variable vector, i.e., set of parameters that determine behavior of the system and are to be manipulated during the optimization process

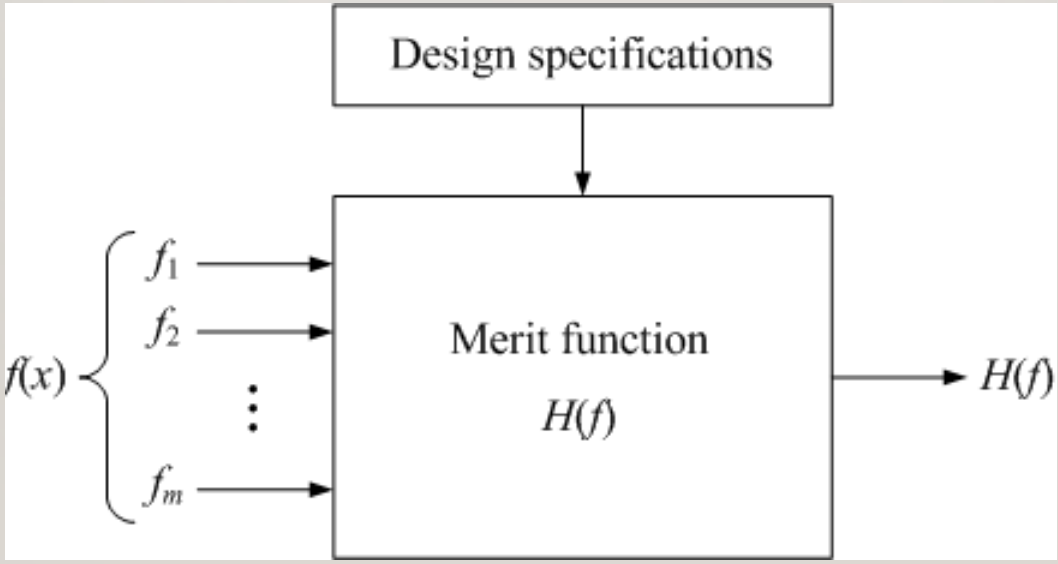
X – domain of f, i.e., the set of feasible design variable vectors

$f = [f_1\ f_2\ \ldots\ f_m]^T$ Î R^m – system response, i.e., set of (output) parameters of the system that are relevant from the point of view of the desired system behavior

Engineering Optimization Problem Formulation

Design specifications: desired behavior of the system expressed in terms of its response

Merit function: a scalar function *H* that “measures” the quality of fulfilling the design specifications by the system response



Typically, better fulfillment of the design specifications translates into smaller values of the merit function

Examples of Merit Functions: *L*-square (L2-norm)

Let $f = [f_1\ f_2 \ldots f_m]^T$ denote the model response components

Specification vector: $S = [S_1\ S_2 \ldots\ S_m]^T$

We want to minimize distance between f and S

The merit function is defined by the norm

$$H(f) = \| f - S \| = \sqrt{\sum_{k=1}^{m} (f_k - S_k)^2}$$

Examples of Merit Functions: Minimax (L∞-norm)

Let $f = [f_1\ f_2\ \ldots\ f_m]^T$ denote the model response components

Lower specification vector: $S_l = [S_{l.1}\ S_{l.2}\ \ldots\ S_{l.m}]^T$

Upper specification vector $S_u = [S_{u.1}\ S_{u.2}\ \ldots\ S_{u.m}]^T$

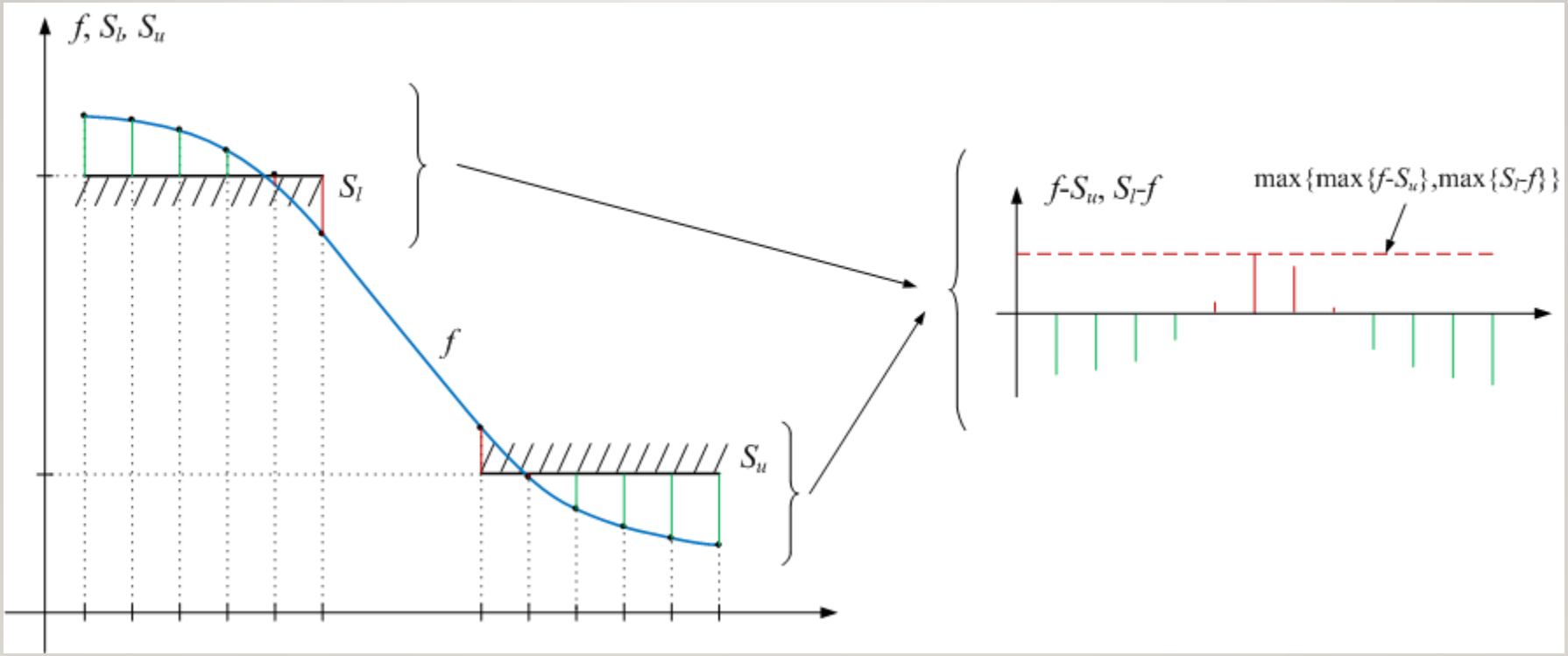
We require that $f_i \leq S_{u.i}$ for $i \in I_u$ and $f_i \geq S_{l.i}$ for $i \in I_l$, where $I_l, I_u \subset \{1, 2, ..., m\}$

The minimax merit function is defined as

$$H(f) = \max\left\{\max_{i \in I_u}(f_i - S_{u.i}), \max_{i \in I_l}(S_{l.i} - f_i)\right\}$$

Examples of Merit Functions: Minimax (L∞-norm)

Illustration of the minimax merit function:



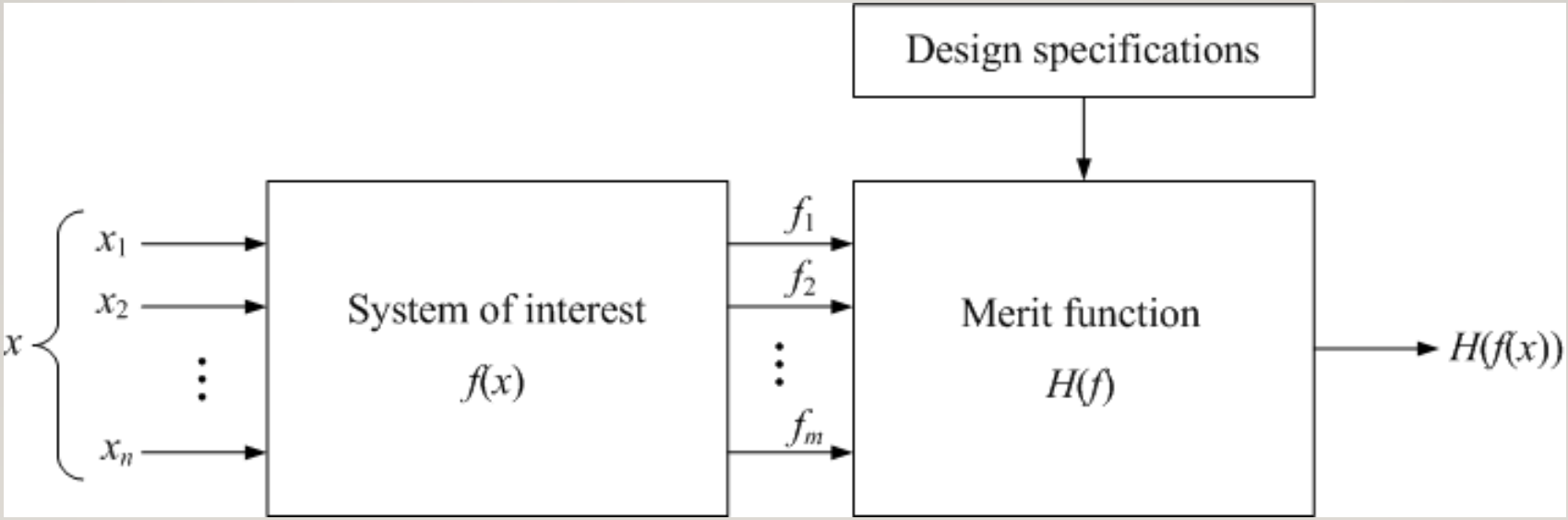
Engineering Optimization Problem Formulation

Composition of H and f, i.e., $H(f(.))$ is called the objective function (cost function)

We want to solve the following problem

$$x^* \in \arg\min_{x \in X} H(f(x))$$

i.e., find x^* that minimizes $H(f(.))$ over X



Engineering Optimization Problem Formulation

Alternative formulation: solve

$$x^* \in \arg\min_{x \in R^n} H(f(x))$$

subject to

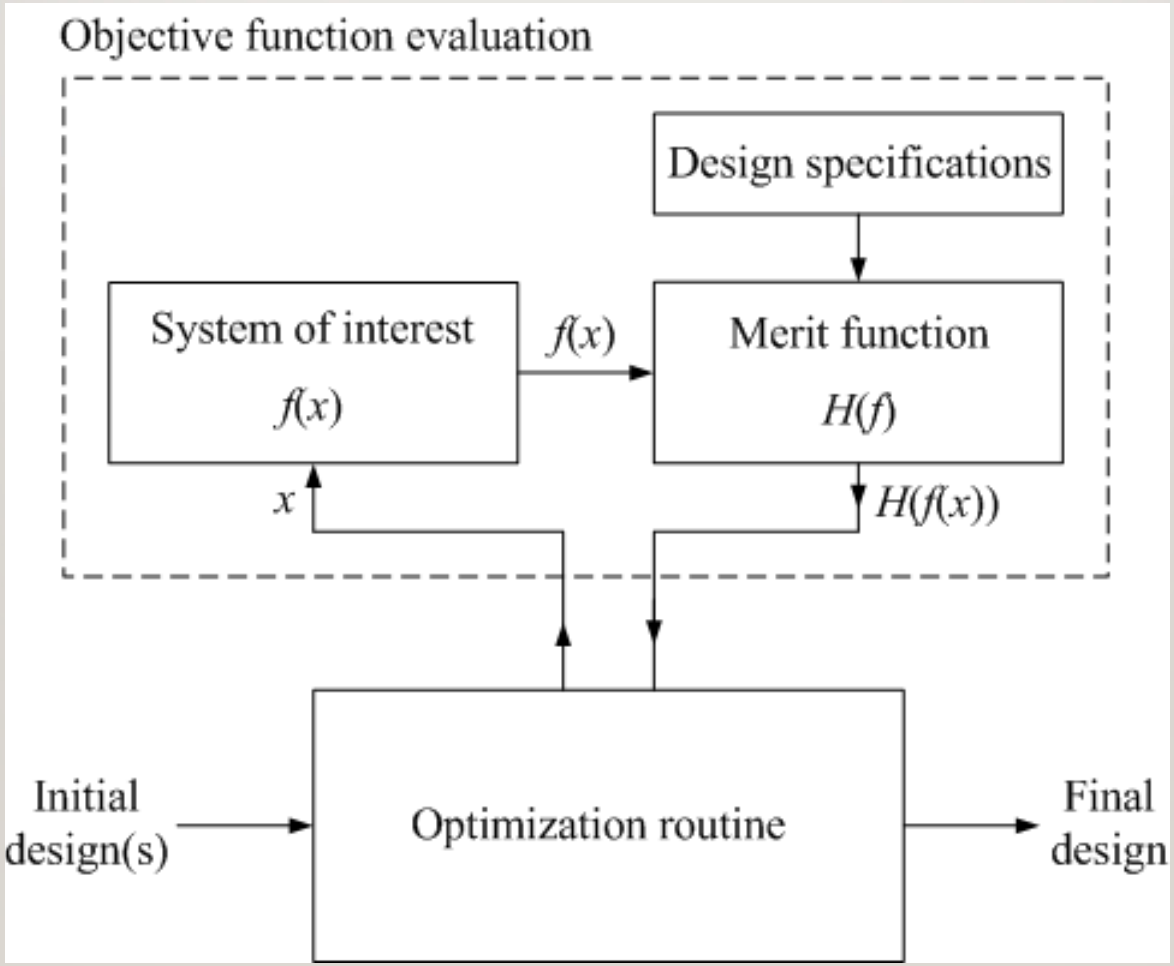
$$c_{eq.k}(x) = 0 \qquad k = 1, 2, ..., n_{eq} \qquad \text{(equality constraints)}$$

and

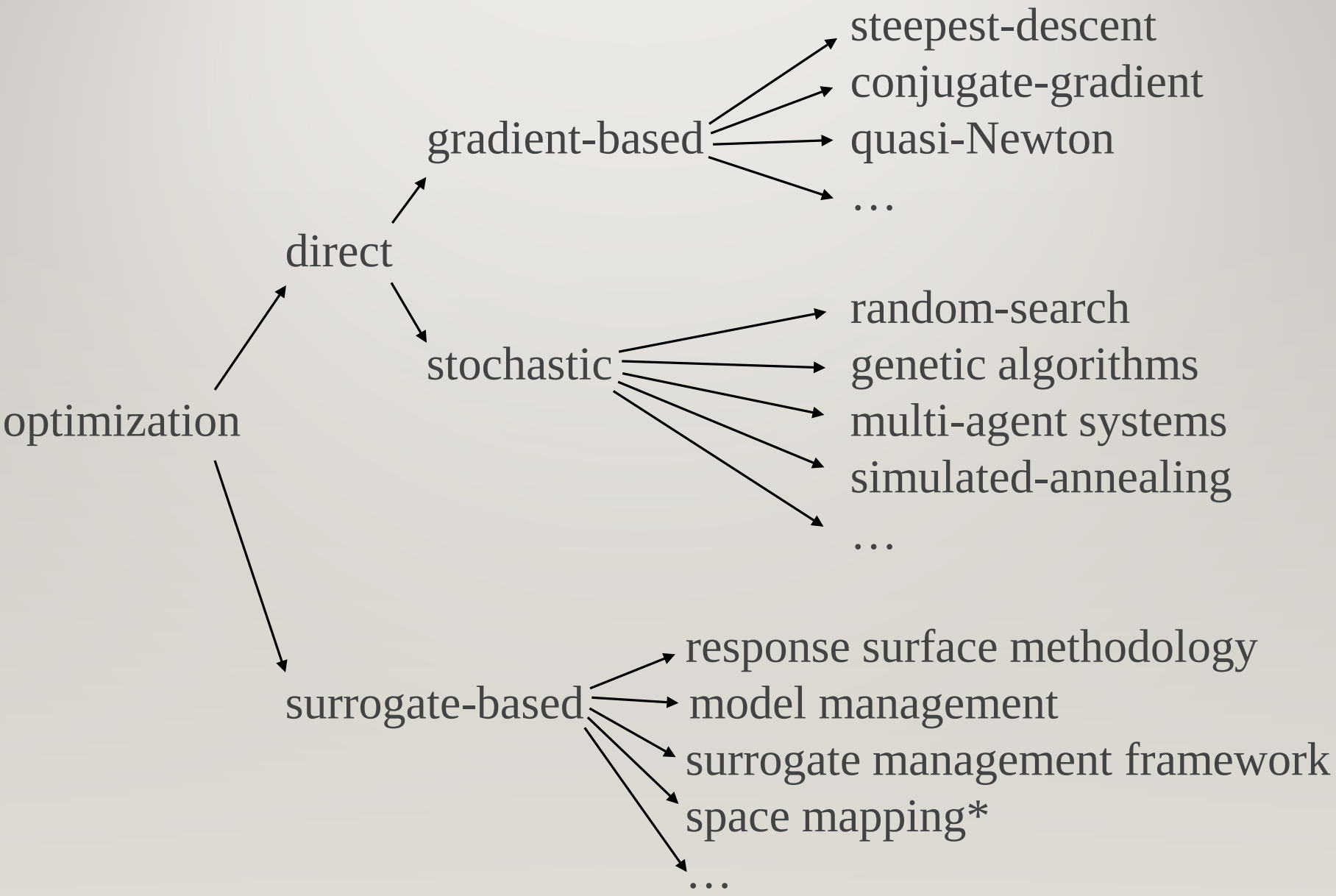
$$c_{ineq.k}(x) \leq 0 \qquad k = 1, 2, ..., n_{ineq} \qquad \text{(inequality constraints)}$$

This formulation is used in practical applications because it allows us to determine whether a given *x* is **feasible** or not

Engineering Optimization Process Flow



Optimization Methodologies



Choosing Optimization Methodology

Optimization methodology is chosen based on the following criteria:

1. Analytical properties of the objective function and constraints (continuity, smoothness, convexity)

2. Availability of derivatives

3. Evaluation cost of the objective function

4. Other factors (e.g., multiple local optima, noisy data, continuous/discrete domain, etc.)

Gradient-based Optimization Methods

Normally require derivative of the objective function (gradient) to proceed; sometimes higher order derivatives are also used

Can find local optimum of the objective functions

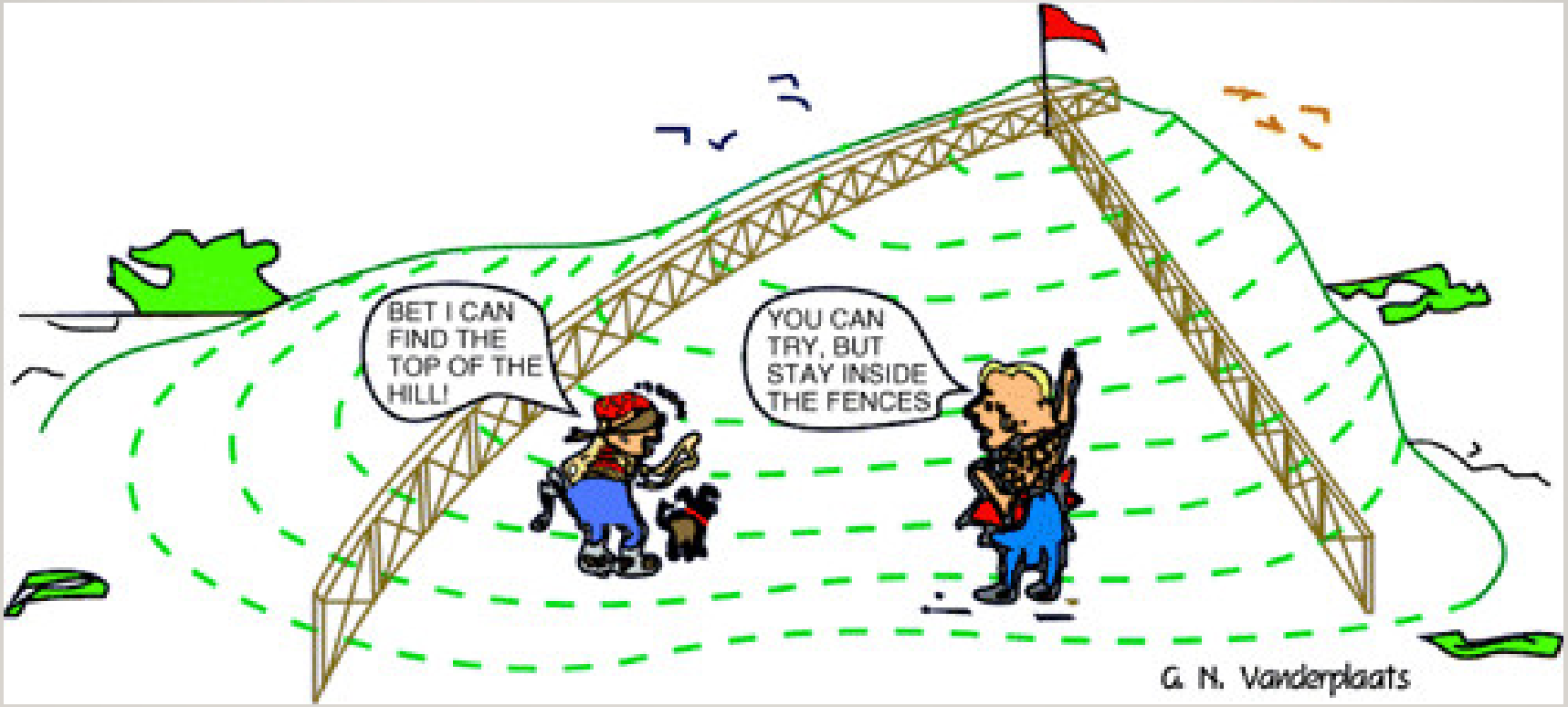
Dependent on the initial solution (or initial design)

Well-established, classical methods with good convergence properties

Not suitable for problems with discontinuous and non-differentiable objective functions

Not suitable for problems with multiple local optima, computationally expensive objective function/constraints, nor the cases where objective function derivatives are not available

Gradient-based Optimization Methods



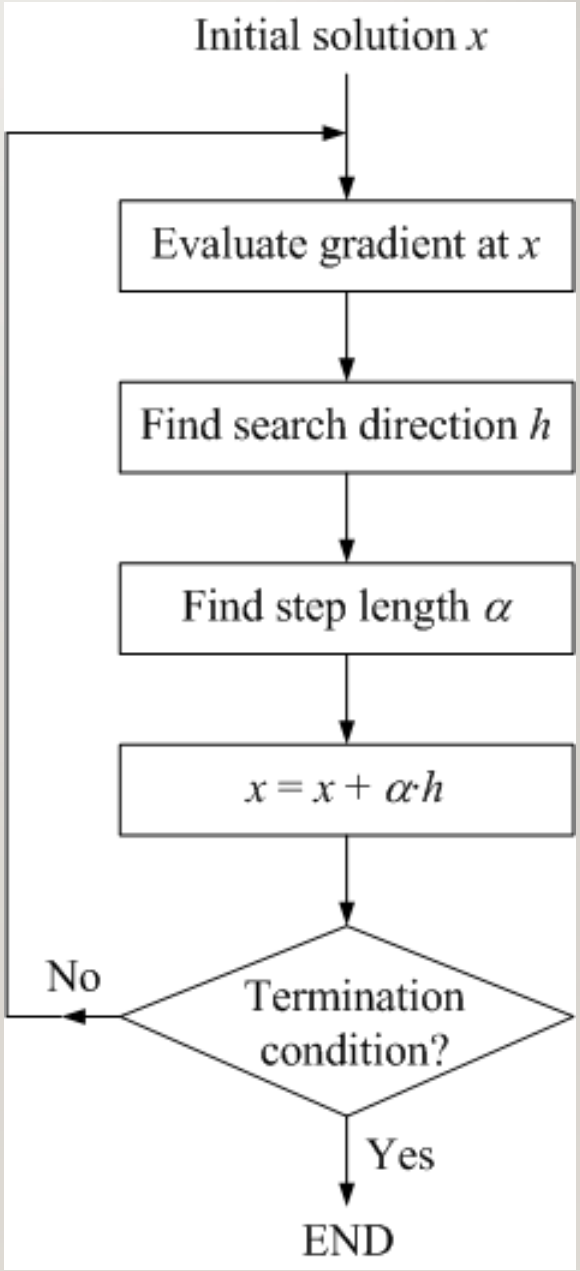
Example: Descent Methods

In each iteration, descent methods find the search direction along which the objective function is (locally) decreasing

Objective function is minimized along the search direction

Design x is updated accordingly

Algorithm is terminated when no further improvement is possible or maximum number of function evaluations is exceeded



Stochastic Optimization Methods

Normally do not require derivative of the objective function; can handle discontinuous and non-differentiable functions

To some extent using element of randomness; many of the methods mimic behavior of natural organisms

Theoretically able to find global optimum of the objective function

Less dependent on the initial solution (design)

Convergence and repeatability of solution not guaranteed in general

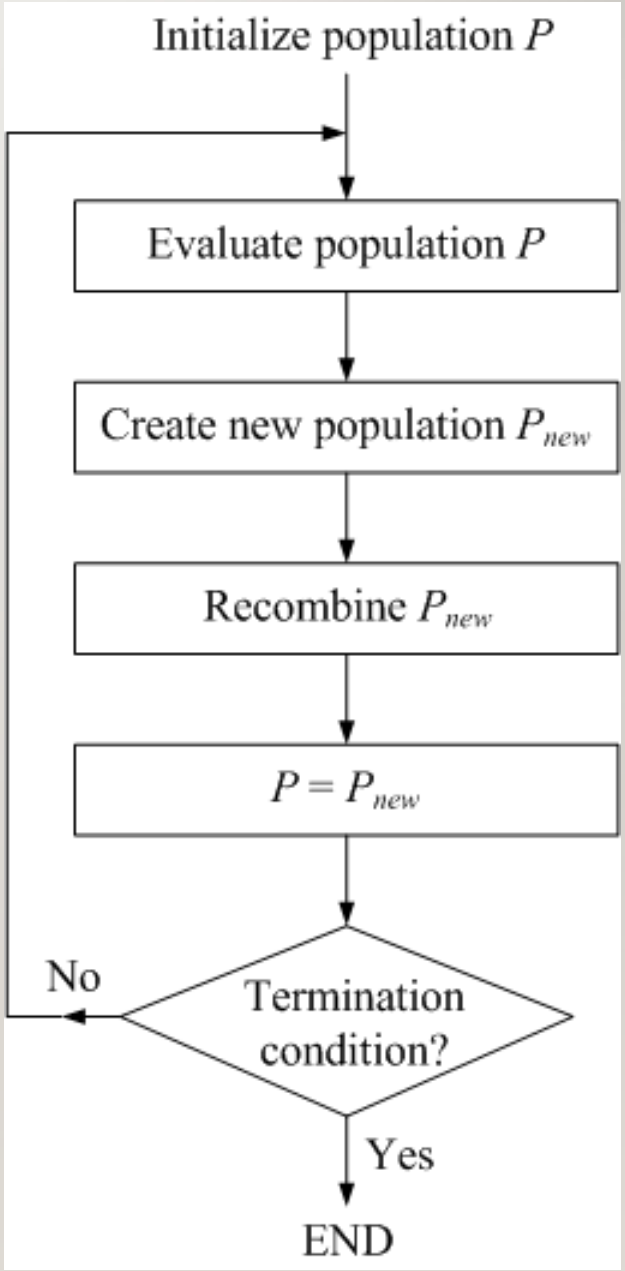
Typically require significant number of objective function evaluations, therefore not suitable for problems with computationally expensive objective function/constraints

Example: Genetic Algorithms

In each iteration, population of designs (individuals) is processed

An updated population is created using the individuals selected from the existing one, then it is recombined using certain genetic operators such as crossover and mutation

Genetic operators use element of randomness but also objective function value so that the best individuals have greater chance to transfer their data into the new population



Genetic Algorithms: Learn to Jump over Ball

Surrogate-based Optimization Methods

Typically do not require derivative of the objective function

Do not work directly on the objective function but create and update so-called surrogate model of the objective function instead

Main optimization steps are performed on the surrogate model; original objective function is referred to for verification purposes

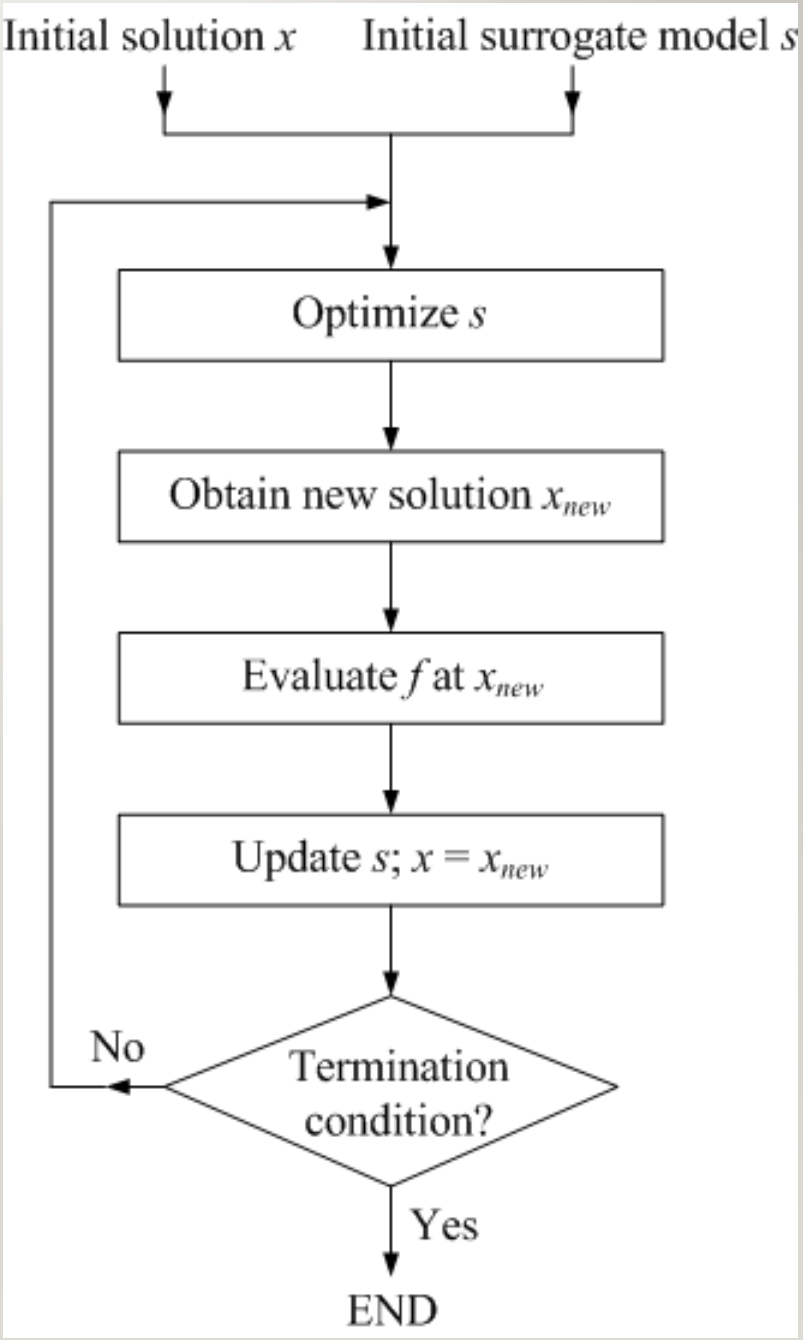
Typically require small number of objective function evaluations, therefore suitable for problems with computationally expensive objective function/constraints, lack of derivative information, and noisy data

Surrogate-based Optimization Flowchart

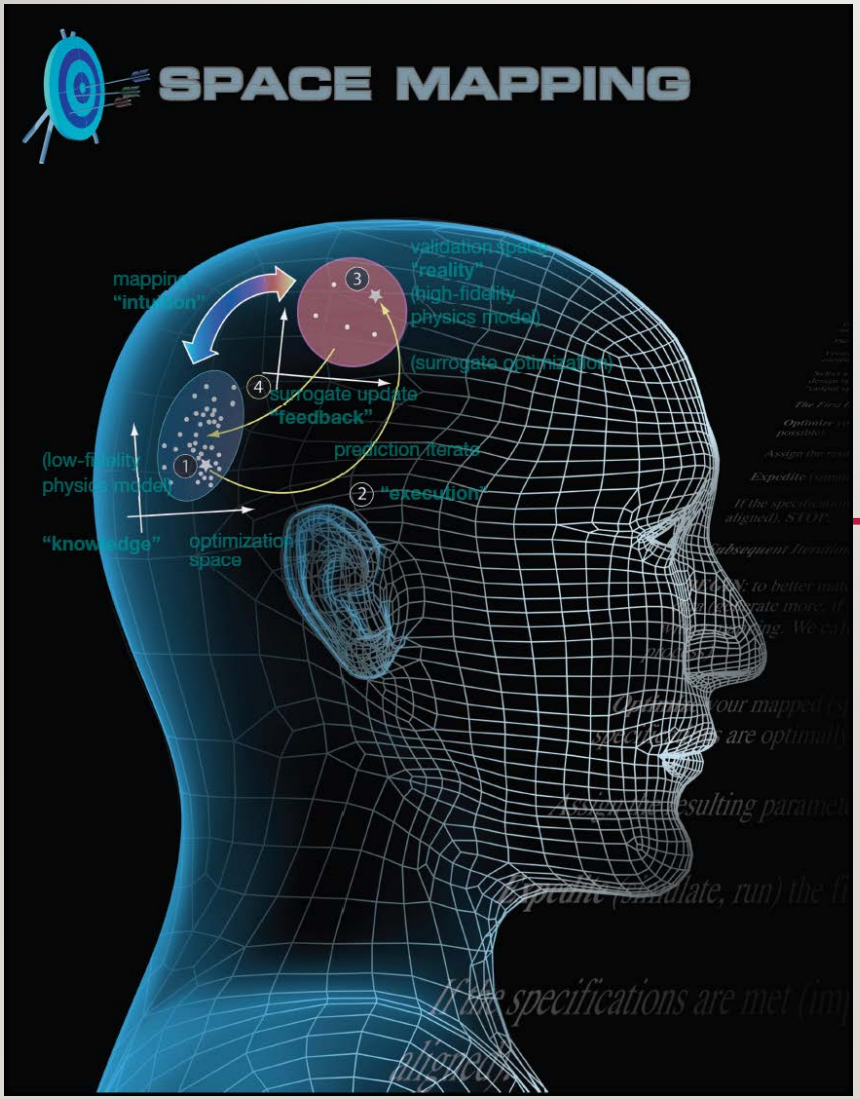
In each iteration, surrogate model of the objective function is optimized to obtain new approximation of the optimal design

New solution is verified using the original objective function information

Surrogate model is updated using the objective function data accumulated during the iteration

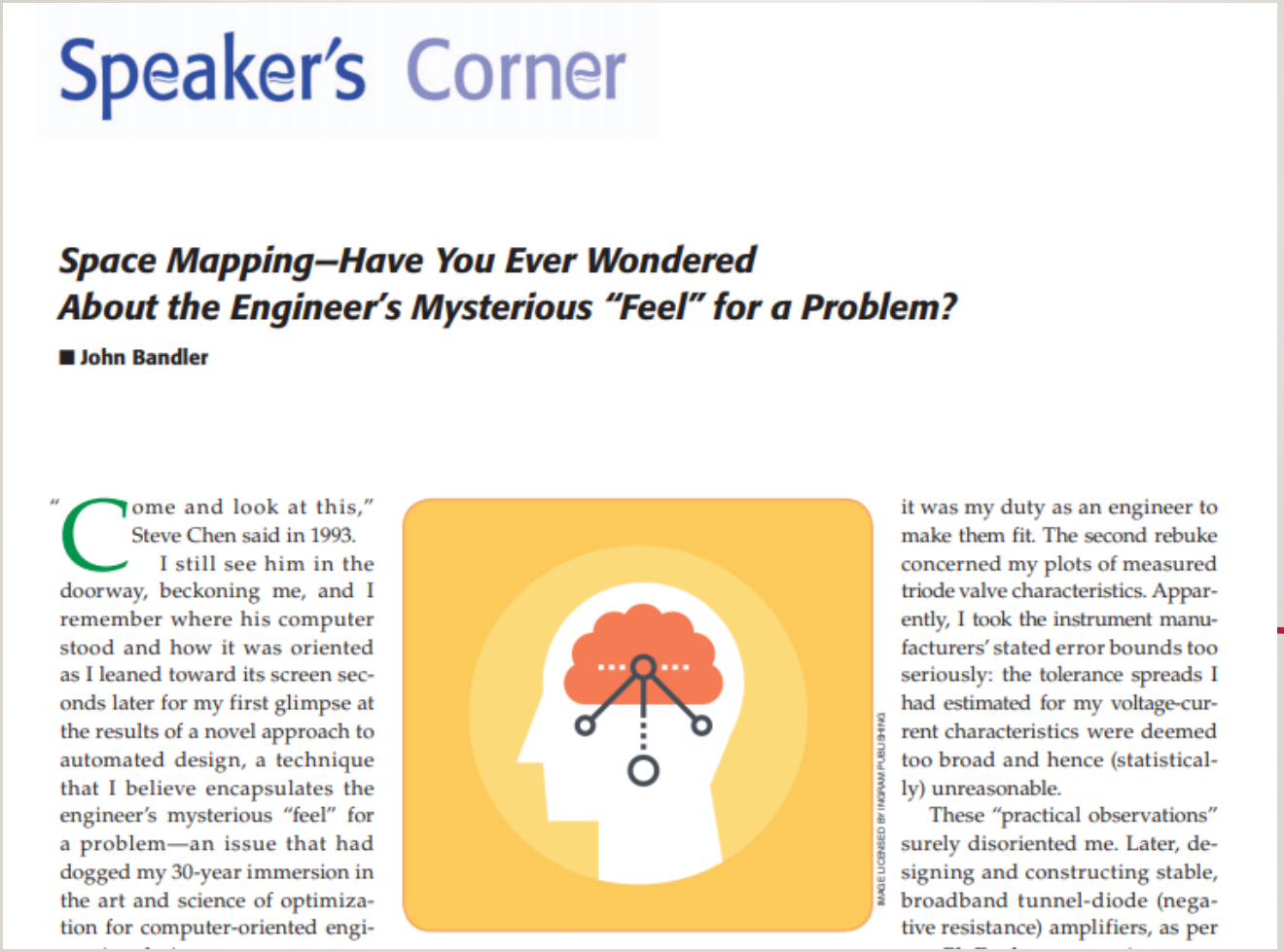


Space Mapping: A Surrogate-Based Optimization Method Use Physics Based Model as Surrogate



J.W. Bandler, "Have you ever wondered about the engineer's mysterious 'feel' for a problem?" Feature Article, *IEEE Canadian Review*, no. 70, pp. 50-60, Summer 2013.

Space Mapping: The Engineer's Mysterious "Feel"



Speaker's Corner

Space Mapping—Have You Ever Wondered About the Engineer's Mysterious "Feel" for a Problem?

■ John Bandler

"Come and look at this," Steve Chen said in 1993. I still see him in the doorway, beckoning me, and I remember where his computer stood and how it was oriented as I leaned toward its screen seconds later for my first glimpse at the results of a novel approach to automated design, a technique that I believe encapsulates the engineer's mysterious "feel" for a problem—an issue that had dogged my 30-year immersion in the art and science of optimization for computer-oriented engi-

it was my duty as an engineer to make them fit. The second rebuke concerned my plots of measured triode valve characteristics. Apparently, I took the instrument manufacturers' stated error bounds too seriously: the tolerance spreads I had estimated for my voltage-current characteristics were deemed too broad and hence (statistically) unreasonable.

These "practical observations" surely disoriented me. Later, designing and constructing stable, broadband tunnel-diode (negative resistance) amplifiers, as per

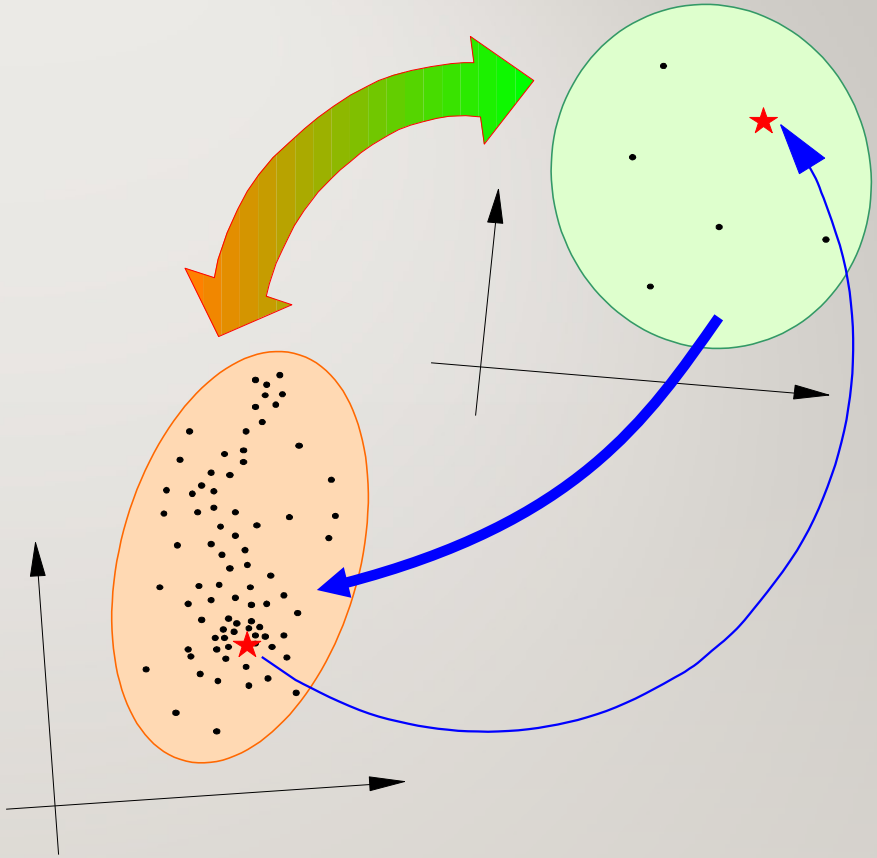
J.W. Bandler, "Space mapping—have you ever wondered about the engineer's mysterious 'feel' for a problem?" Feature Article, IEEE Canadian Review, no. 70, pp. 50-60, Summer 2013. Reprinted in *IEEE Microwave Magazine*, vol. 19, no. 2, Mar./Apr. 2018.

Space Mapping
(*Bandler et al., 1994*)

follows common sense
knowledge and experience

exploits the common sense
"intuition"—match and predict

uses an iterative approach
to interact with reality



"space mapping" offers
a quantitative explanation for
performing common sense activities

Exercise 1: Travelling Salesman Problem

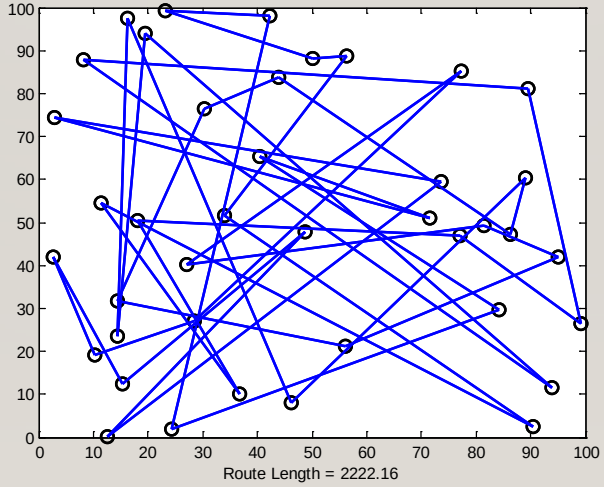
There are N cities with the distance between city i and city j given by d_{ij}. Problem: find a close route through all the cities so that minimizes the total route length.

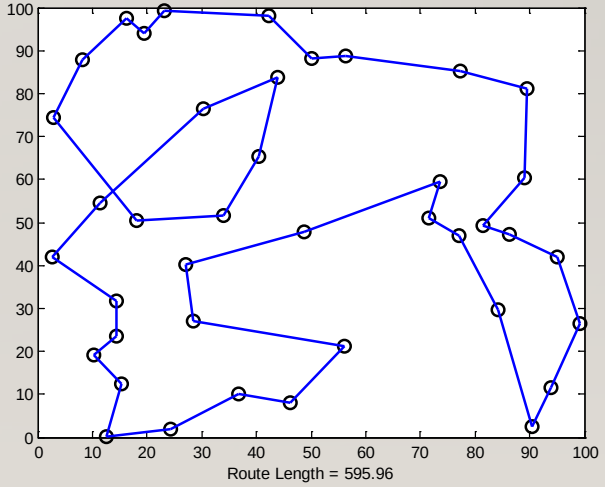
Write a Matlab function that implements the objective function for this problem. The input arguments are a route described as a permutation of the numbers 1 to N, as well as the distance data stored in an appropriate form.

Demonstrate operation of your function using examples of your choice. Visualize the cities location as well as a route using suitable plots (see below).

Implement and visualize simple optimization procedure that modifies the route by random swapping two cities on the route (the updated route is accepted if it reduces the objective function).

TSP for 40 cities:
Initial route (left)
Optimized route (right)

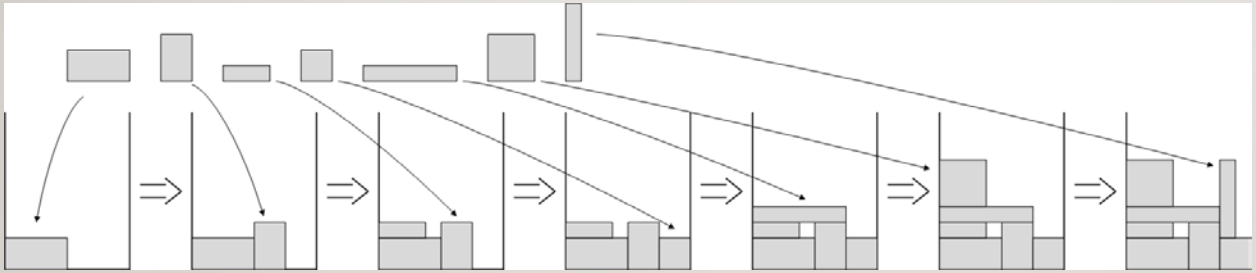




Exercise 2: Bin Packing Problem

Problem: Given a 2D bin of width W, allocate N rectangles of sizes $w_k \times h_k$ so that the height of the packed rectangles is minimal.

Write a Matlab function that given an order of the rectangles to be packed and their orientation, puts them from left to right and as low as possible (see below).



Implement a simple algorithm that randomly swaps the order of two rectangles and (randomly) changes their orientation (the updated order is accepted if it reduced the objective function value). Visualize the process.

Bin packing problem for 40 rectangles (W = 12).
Initial order (left) and optimized order (right)